



BazelCon 2026



MEROTON
SPEED UP YOUR BUILDS

Discovering and eliminating 20% of redundant build actions

by tracing file system operations



Oskar Sjöberg



Fredrik Medley

#BazelCon



Problem statement

Undeclared inputs => Build failure

Overdeclared inputs => Waste

- Cache keys depend on *all* inputs, even if they are unused
- Rebuilding when unused inputs are changed



Questions

How much waste?
And how do we reduce it?



BazelCon 2026



MEROTON
SPEED UP YOUR BUILDS



Example in C/C++

BUILD.bazel

```
cc_library(  
  name = "maths",  
  srcs = [  
    "add.c",  
    "subtract.c",  
  ],  
  hdrs = [  
    "add.h",  
    "subtract.h",  
  ],  
)
```

add.h

```
int add(int x, int y);
```

add.c

```
#include "add.h"  
int add(int x, int y)  
{  
    return x + y;  
}
```

subtract.h

```
int subtract(int x, int y);
```

subtract.c

```
#include "subtract.h"  
int subtract(int x, int y) {  
    return x - y;  
}
```

This is an example, our solution is language- and rule-agnostic.

System setup: Remote execution with Buildbarn.



bazel aquery

BUILD.bazel

```
cc_library(  
  name = "maths",  
  srcs = [  
    "add.c",  
    "subtract.c",  
  ],  
  hdrs = [  
    "add.h",  
    "subtract.h",  
  ],  
)
```

```
$ bazel aquery 'mnemonic("CppCompile", ":maths")'
```

```
action 'Compiling add.c'
```

```
  Inputs: [add.c, add.h, subtract.h]
```

```
action 'Compiling subtract.c'
```

```
  Inputs: [add.h, subtract.c, subtract.h]
```



BazelCon 2026



MEROTON
SPEED UP YOUR BUILDS



bazel build

```
$ bazel build --subcommands :maths
```

Compiling add.c

Compiling subtract.c

INFO: 5 processes: 1 internal, 4 remote.

File access are logged on file system level
using a virtual file system.

Buildbarn Portal☰

Input files

▼ > ≡ >>

📁 action_pruning	393 B	drwxr-xr-x
📄 add.c	62 B	-r-xr-xr-x
📄 add.h	37 B	-r-xr-xr-x
📄 subtract.h	51 B	-r-xr-xr-x
📁 bazel-out	1.14 kiB	drwxr-xr-x
📁 external	>101 MiB	drwxr-xr-x

Implementation

After execution

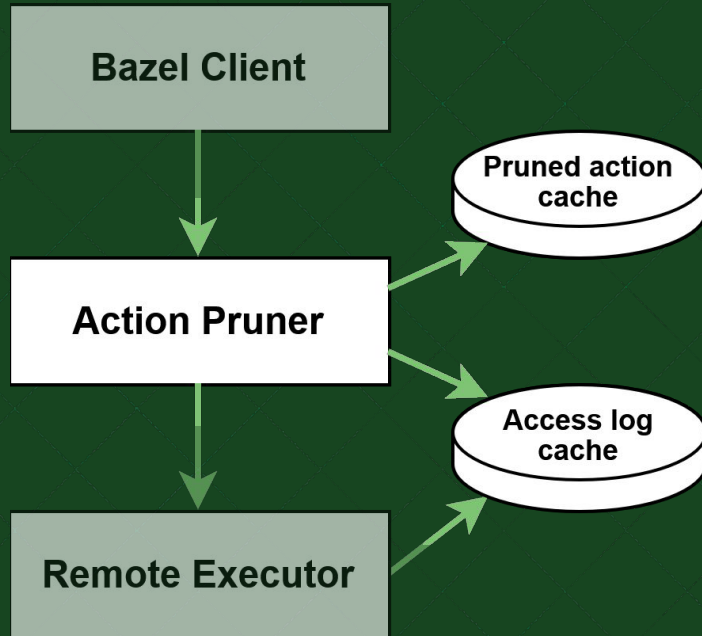
Store extra cache entry

1. Prune inputs using file system access log
2. Cache results

Before execution

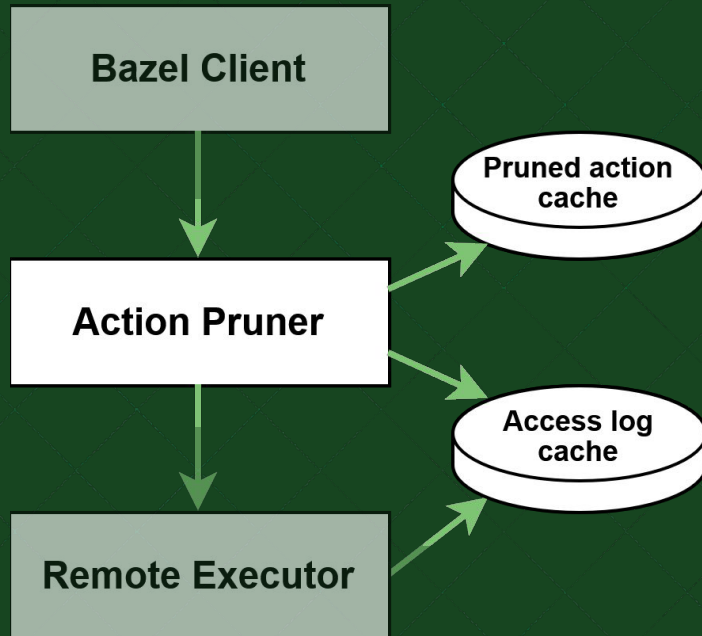
Extra cache lookup

1. Prune previously unused inputs
2. Lookup in cache



Similar actions

- **Full action** to execute
 - Remote Execution API v2
- **Pruned action** for action cache lookups
 - Need file access log for pruning.
- **Reduced action** for file access logs

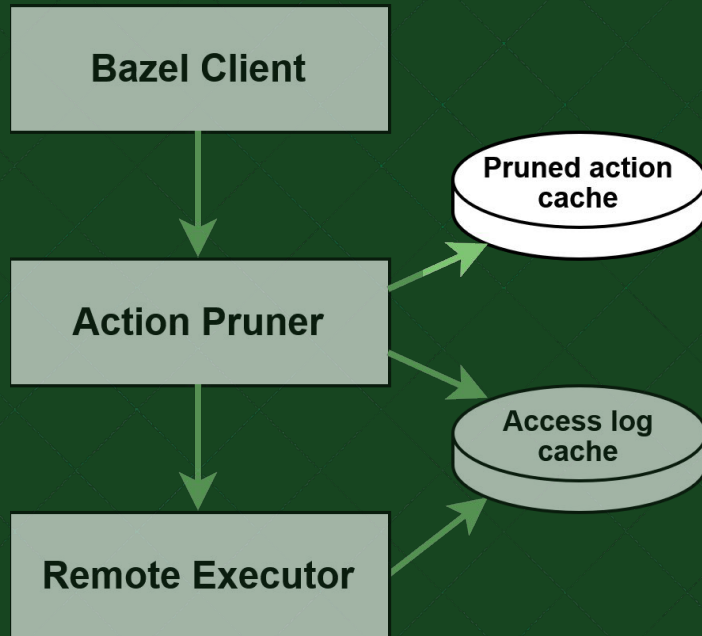


Pruned action

Action in Remote Execution API v2

- Command
 - `gcc add.c`
- Platform properties
 - `Linux`
- Input tree
 - `add.c`, `add.h`, ~~`subtract.h`~~

Unused files are removed.



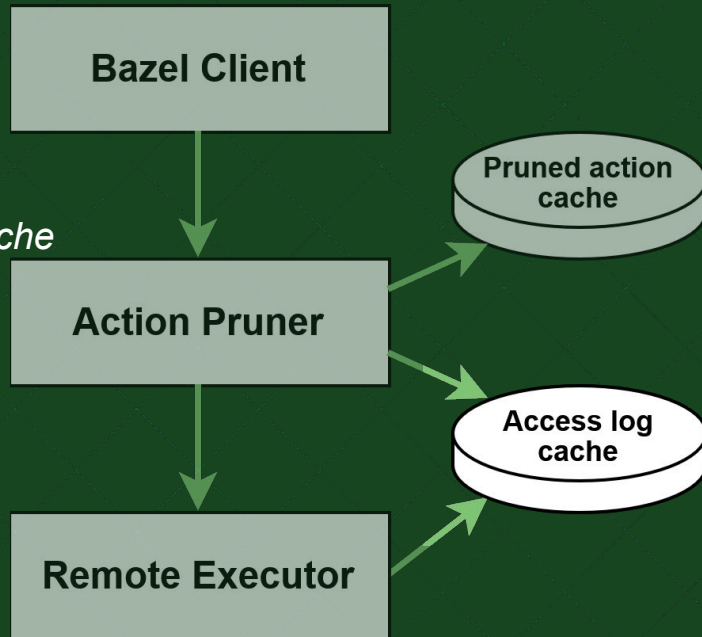
Reduced action

Action in Remote Execution API v2

- Command
 - `gcc add.c`
- Platform properties
 - `Linux`
- Input tree
 - `add.c, add.h, subtract.h`

Key in *file system access cache*

The command contains paths to the input and output files, identifies a specific action for a target.



Alternatives - .d file pruning

Bazel's built-in .d file pruning

- Only works for C/C++
- Analysis cache (local host)

```
$ bazel build --subcommands :maths
Compiling add.c
Compiling subtract.c
INFO: 5 processes: 1 internal, 4 remote.
```

```
$ cat bazel-out/.../add.pic.d
bazel-out/k8-fastbuild/bin/_objs/maths/add.pic.o: \
    add.c \
    add.h
```

```
# Modify subtract.h
```

```
$ bazel build --subcommands :maths
Compiling subtract.c
INFO: 2 processes: 1 action cache hit, ..., 1 remote.
```

Generic interface

```
ctx.actions.run(unused_inputs_list)
```

File containing list of inputs unused by the action. [...] Any change in those files must not affect in any way the outputs of the action.

Alternatives - IWYU and DWYU

Useful linters for C/C++ code.

Include **What You Use**

- Avoid including headers that are not used

Depend on **What You Use**

- Avoid listing unused **deps**

add.h

```
int add(int x, int y);
```

add.c

```
#include "add.h"
int add(int x, int y)
{
    return x + y;
}
```



Results

C++ project at Volvo Cars Corporation

Millions actions analysed

One in five were wasteful



20%



Discovering and eliminating 20% of redundant build actions by tracing file system operations

Feedback link





BazelCon 2026



MEROTON
SPEED UP YOUR BUILDS

Thank you for listening



Discovering and eliminating 20% of redundant build actions by
tracing file system operations

#BazelCon

